Date __/__/__ HTML and CSS assignment. Page No.:

① What is the CSS box model, and how does it affect the layout of elements on a webpage?

→ The CSS box model is a fundamental concept that defines the structure of every HTML element on a webpage. It consists of content, padding, border, and margin.

- Content:→ the actual element's content (text, images, etc.).
- Padding:→ Clears space between the content and the border.
- Border:→ A border surrounding the padding (if any).
- Margin:→ Clears space outside the border, separating it from other elements.

These components collectively determining the size and spacing of an element, influencing its layout on the page. Adjusting these properties can impact how elements are positioned relative to each other and their surrounding environment.

Exercise:- Provide a simple HTML structure with a few elements and ask the students to apply CSS properties to manipulate the box model, such as margin, padding, and borders.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width
    = device-width, initial-scale=0">
```

Date __/__/____ Page No.:_____

```
<title> Box Model Exercise </title>
<style>
 body {
    font-family : Arial, sans-serif;
    margin : 20px;
 }

 .container {
    background-color : #f0f0f0;
    padding : 10px;
    border : 2px solid;
    margin : 10px;
 }

 .box1 {
    background-color : #ffd700;
    padding : 15px;
    border : 1px solid #00;
    margin : 10px;
 }

 .box2 {
    background-color : #90ee90;
    padding : 20px;
    border : 1px solid #008000;
    margin : 10px;
 }
</style>
</head>
<body>
    <div class = "container">
```

Date __/__/____ Page No.:_____

```
<h1> welcome to Box Model Exercise </h1>
<p> Modify the CSS properties to manipulate the
    box model of the following boxes : </p>
<div class = "box 1">
<p> This is Box 1 </p>
</div>
<div class = "box2">
   <p> this is Box 2 </p>
</div>
</div>

</body>
</html>
```

in this Example, Students can Experiment with adjusting the margin, padding and border properties for the '.container', '.box1', and '.box2' class to observe the Effects on the layout of the page.

(2) Explain the concept of CSS Specificity. How is it determined and why is it important in styling web pages?

→ CSS specificity is a set of rules that determine which styles are applied to an element when conflicting styles are present. It's a way for browsers to decide which style declarations should take precedence. Specificity is calculated based on the combination of selectors used to target an element. four components:-

(1) Inline Styles: styles applied directly to an element with the 'style' attribute have the

Date __/__/____ Page No.:_____

highest specificity.

② IDs: Selectors containing ID attributes have higher specificity than classes or tag selectors.

③ Classes, attributes, and pseudo-classes: These have a lower specificity compared to IDs but are higher than tag selectors.

④ Element type: Selecting by element types (e.g., `div`, `p`) has the lowest specificity.

If two conflicting styles have the same specificity, the one declared later in the stylesheet takes precedence due to the principle of the "cascading" nature of stylesheets.

Understanding specificity is crucial in web development because it helps developers predict and control which styles will be applied to an element. It prevents unexpected styling conflicts and ensures that styles are applied in a logical and predictable manner, making it easier to maintain and troubleshoot stylesheets.

Exercise → Present a set of HTML elements and CSS rules with varying levels of specificity, and ask student to predict the final styles applied to each element.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-
    width, initial-scale=1.0">
```

Date __/__/____ Page No.:_____

```
<title> Specificity Challenge </title>
<link rel="stylesheet" href="styles.css">
<style>                    /* Rule 4 */
  div {
    color : blue;
  }
</style>
</head>
<body>
  <div id="element 1" class="box"> Element 1 </div>
  <div class="box"> Element 2 </div>
  <div id="element 3"> Element 3 </div>
</body>
</html>
```

CSS

```
.box {                     /* Rule 2 */
  color : red;
}

#element 1 {               /* Rule 3 */
  color : green;
}

div.box {                  /* Rule 4 */
  color : purple;
}
```

Challenge:

Predict the final text color for "Element 1".

Predict the final text color for "Element 2".

Predict the final text color for "Element 3".

Date __/__/____ Page No.:_____

(3) What are CSS flexbox and CSS Grid? Describe when you would use each layout model, and provide an example of both.

→ CSS flexbox and CSS Grid are layout models in CSS that help building responsive and flexible web design. (with)

CSS flexbox:

- Use case: Ideal for arranging items in a single direction (either horizontally or vertically) and with space distribution within a container.

Example
```
.container {
    display: flex;
    justify-content: space-between;
}
.item {
    flex: 1;
}
```

CSS Grid:

- Use case: Suitable for two-dimensional layout, allowing you to create complex grid-based design with rows and columns.

Example
```
.container {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr;
    grid-gap: 10px;
}

.item {
    grid-column: span 2;
}
```

Date __/__/____ Page No.:_____

In summary, use flexbox for one-dimensional layouts and when dealing with items along a single axis. Use grid for two dimensional layouts., especially when you need to align items in both rows and columns.

Exercise→ Ask students to create a webpage layout using CSS flexbox and another using CSS grid, and compare the differences in the resulting layouts.

CSS flexbox Example:-

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
                                 initial-scale=1.0">
<title>flexbox layout</title>
<style>
  body {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
  }

  .container {
    display: flex;
    justify-content: space-around;
    align-items: center;
    width: 80%;
```

SAMMAN }

Date __/__/____ Page No.:_____

```
.item {
  flex : 1;
  padding : 20px;
  text-align : Center;
  border : 1px solid #ccc;
  margin : 5px;
}

.item {
  flex : 1;
  padding : 20px;
  text-align : center;
  border : 1px solid #ccc;
  margin : 5px;
}
  </Style>
</head>
<body>
  <div class = "container">
    <div class = "item" > Item 1 </div>
    <div class = "item" > Item 2 </div>
  </div>
</body>
</html>
```

CSS Grid Example:

```
<!DOCTYPE html>
<html lang = "en">
</head>
  <meta charset = "UTF-8">
  <meta name = "viewport" content = "width=device-width
      ,initial-scale = 1.0">
```

Date __/__/____ Page No.:_____

```
<title> Grid layout </title>
<style>
  body {
    display : grid;
    place-item : center;
    height : 100vh;
    margin : 0;
  }

.container {
    display : grid;
    grid-template-columns : repeat (3, 1fr);
    gap : 10px;
    width : 80% ;
  }

.item{
  padding : 20px;
  text-align : center;
  border : 1px solid #ccc;
}
</style>
</head>
<body>
<div class = "container">
    <div class = "item" > item 1 </div>
    <div class = "item" > item 2 </div>
    <div class = "item" > item 3 </div>
  </div>
</body>
</html>
```

Date __/__/__ Page No.:_____

Comparison:

- Flexbox is well-suited for distributing items along a single axis (either horizontally or vertically), while Grid provides a two-dimensional layout system, allowing control over both rows and columns.
- Flexbox is more suitable for one-dimensional layouts, like navigation bars, whereas Grid is powerful for complex, two-dimensional layouts, such as a complete webpage structure.
- Flexbox uses 'flex' property to distribute space, while Grid uses 'grid-template-columns' and 'grid-template-rows'.
- Flexbox is often used for components within a layout, while Grid is often used for defining the overall layout structure.

④ Describe the difference between 'position: relative', 'position: absolute', and 'position: fixed' in CSS. When and how would you use each of these position values?

→ (1) 'position: relative':

- Elements with 'position: relative' are positioned relatives to their normal position in the document flow.
- If you move a 'relative' element using properties like "top", 'right', 'bottom', or 'left', it will be adjusted from its normal position.
- This is often used for minor adjustments within a page layout.

Date __/__/____ Page No.:_____

(2) 'position : absolute':

- Elements with "position : absolute" are positioned relative to the nearest positioned ancestor (an ancestor with a position other than 'static'), if there is one. If not, it's positioned relative to the initial containing block (often the '<html>' element).
- The element is taken out of the normal flow of the document, meaning it won't affect the layout of other elements.
- This is commonly used for creating overlays, pop-ups, or positioning elements precisely within a container.

(3) 'position : fixed':

- Elements with 'position : fixed' are positioned relative to the viewport.
- These elements stay fixed even when the page is scrolled. They do not move with the rest of the document.
- Commonly used for headers, footers, or any element that you want to stay visible as the user scrolls.

When to use each:

- Use 'relative' for minor adjustments within the normal flow of the document.
- Use 'absolute' when you want an element taken out of the normal flow and positioned relative to a parent container or the document itself.

SAMMAN

Date __/__/____ Page No.:_____

- Use 'fixed' when you want an element to stay in a fixed position relative to the viewport regardless of scrolling.

Exercise: Provide a webpage with elements that need to be positioned using the different values, and ask students to apply CSS to achieve the desired layout.

```
<!DOCTYPE html>
<head>
  <meta charset = "UTF-8">
  <meta name = "viewport" content = "width = device-width,
        initial-scale = 1.0">
  <title> Positioning Example </title>
  <style>
    body {
      margin : 0;
      padding : 20px;
      font-family : Arial, sans-serif;
    }
    .relative-container {
      position : relative;
      height : 200px;
      background-color : #f0f0f0;
      margin-bottom : 20px;
    }
    .absolute-element {
      position : absolute
      top : 10px;
      left : 10px;
      background-color : #3498db;
```

```
    color: #fff;
    padding: 10px;
}

.fixed-element{
    position: fixed;
    top: 10px;
    right: 10px;
    background-color: #e74c3c;
    color: #fff;
    padding: 10px;
}

.normal-flow-element{
    background-color: #2ecc71;
    color: #fff;
    padding: 10px;
    margin-top: 250px;
}
</style>
</head>
<body>
    <div class="relative-container">
        <div class="absolute-element">Absolute</div>
        <div class="fixed-element">Fixed element</div>
        <div class="normal-flow-element">normal flow</div>
    </div>
<p>This is some regular content in the normal
    flow of the document</p>
</body>
</html>
```

Date __/__/____ Page No.:_____

Ques- Explain the concept of CSS pseudo-elements like 'before' and 'after'. provide an example where these pseudo-elements are used to enhance the design of a webpage.

CSS pseudo-elements like 'before' and 'after' are used to insert content before or after an element's actual content, without the need for additional HTML. They're often used to enhance design or add decorative elements.

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-
        width, initial-scale=1.0">
  <link rel="stylesheet" href="styles.css">
  <title>Page with Pseudo-Elements</title>
</head>
<Body>
  <div class="container">
    <h1> Hello, World! </h1>
    <p> This is a Sample text </p>
  </div>
</body>
</html>
```

Now, you can use 'before' and 'after' to add decorative elements in your 'styles.css':

Date __/__/____ Page No.:_____

CSS-

```
.container::before {
    content: '';
    display: block;
    width: 100%;
    height: 2px;
    background-color: #333;
    margin-bottom: 10px;
}

.container::after {
    content: '★';
    display: block;
    text-align: center;
    font-size: 24px;
    color: #ffcc00;
    margin-top: 10px;
}
```

Give students a webpage with specific design requirements and ask them to use pseudo elements to achieve those enhancements.

→ Here's a simple webpage template with design requirements. Encourage students to use CSS pseudo-elements '::before' and '::after' to achieve the specified enhancements:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-
                width, initial-scale=1.0">
```

Date __/__/____ Page No.:_____

```
<link rel="stylesheet" href="styles.css">
<title>Enhance this page</title>
</head>
<body>
  <header>
    <h1>welcome to our page</h1>
  </header>

  <section class="main-content">
    <p>This is some main content. Make it
    visually appealing with pseudo-elements.</p>
  </section>

  <footer>
    <p>contact us at : contact@Example.com</p>
  </footer>
</body>
</html>
```

CSS →

```
header::before {
    content: '';
    display: block;
    height: 10px;
    background-color: #3498db;
}

.main-content::after {
    content: '\2022';
    color: #e74c3c;
    font-size: 24px;
    margin-top: 10px;
```

SAMMAN

```
    display : block;
}

footer ::before {
    content : '✆';
    margin-right : 5px;
}
```

in this Example:

- 'header ::before' adds a blue bar above the header.
- '.main-content ::after' adds a red bullet point after the main content.
- 'footer::before' adds a phone emoji before the contact information in the footer.

Ques 6→ What is responsive web design, and how can media queries be used to create responsive layouts? Provide an example of a responsive webpage.

Responsive web design is an approach to designing and coding websites to ensure that they look across various devices and screen sizes. The goal is to create a seamless user experience, optimizing the layout and content for different devices such as desktops, tablets and smartphones.

Media queries are a key component of responsive web design. They allow you to apply specific CSS styles based on characteristics of the

device, such as its width, height, or device orientation. This enables the adaptation of the device, such as its width, height, or device orientation. This enables the adaptation of the layout and styling to different screen sizes.

Here's a simple example of a responsive webpage using media queries.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width
        , initial-scale=1.0">
    <link rel="stylesheet" href="styles.css">
    <title>Responsive Webpage</title>
</head>
<body>
    <header>
        <h1>My Responsive Webpage</h1>
    </header>

    <section class="main-content">
        <p>This is the main content of the webpage</p>
    </section>

    <footer>
        <p>Contact us at: contact@example.com</p>
    </footer>
</body>
</html>
```

Date __/__/____ Page No.:_____

CSS →

```
body {
    font-family : 'Arial', sans-serif;
    margin : 20px;
}

header, section, footer {
    padding : 20px;
    text-align : center;
}

/* Responsive Styles */
@media only screen and (max-width : 600px) {
    body {
        margin : 10px;
    }

    header, section, footer {
        padding : 10px;
    }
}
```

- The default styles set a margin and padding for the body, and the padding for the header, section and footer.
- The media query targets screens with a maximum width of 600 pixels, and adjusts the styles accordingly, reducing the margin and padding for smaller screens.

SAMMAN

Date __/__/____ Page No.:_____

Provide a basic webpage and ask Students to create a responsive design using media queries to adapt the layout for the different screen sizes.

```
<!DOCTYPE html>
<html lang="En">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
        initial-scale=1.0">
    <title>Responsive Web Design Challenge</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <header>
        <h1> Your Website Name </h1>
        <nav>
            <ul>
                <li><a href='#'> Home </a></li>
                <li><a href='#'> about </a></li>
                <li><a href='#'> Contact </a></li>
            </ul>
        </nav>
    </header>

<main>
    <section>
        <h2> Welcome to our website </h2>
        <p> lorem ipsum dolar sit amet consectetur
            adipiscing elit.... </p>
    </section>
```

```
<section>
  <h2> Latest News </h2>
  <article>
   <h3> News title 1 </h3>
   <p> Short description of the news... </p>
  </article>

  <article>
   <h3> New Title 2 </h3>
    <p> Short description of the news... </p>
  </article>
  </section>
 </main>

 <footer>
   <p> &copy; 2023 Your Website Name. All
     rights reserved. </p>
  </footer>
</body>
</html>
```

CSS →

```
body {
  font-family: Arial, sans-serif;
   margin: 0;
   padding: 0;
}

header {
  background-color: #333;
   color: #fff;
```

Date __/__/____ Page No.:_____

```
        padding : 10px;
        text-align : center;
    }

nav ul {
    list-style : none;
    margin : 0;
    padding : 0;
}

nav li {
    display : inline;
    margin-right : 20px;
}

main {
    padding : 20px;
}

footer {
    background-color : #333;
    color : #fff;
    text-align : center;
    padding : 10px;
    position : absolute;
    bottom : 0;
    width : 100%;
}
```

Page No.:

Date __/__/____

Ques → Discuss the importance of accessibility in web development. Explain ARIA roles and attributes, and provide an example of making a webpage more accessible.

Accessibility in web development is crucial as it ensures that people with disabilities can access and interact with websites. It promotes inclusivity and a better user experience for everyone. Accessible websites consider various impairments such as visual, auditory, motor, and cognitive.

ARIA (Accessible Rich Internet Applications) roles and attributes enhance the semantics of web content for assistive technologies. Roles define the type of UI element, and attributes provide additional information. For Example, the 'role' attribute can be set to "button" to indicate that an element functions as a button.

Example →

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
        initial-scale=1.0">
    <title>Accessible Webpage Example</title>
</head>
<body>
    <h1>Accessible Webpage Example</h1>

    <div role="button" tabindex="0" onclick="
```

SAMMAN

Date __/__/____ Page No.:_____

```
handleButtonClick()" onkeypress = "handleButtonKey
Press(event)" aria-label = "click me to perform an action">
Click me
</div>

<script>
  function handleButtonClick(){
      alert ("Button clicked!");
  }

  function handleButtonKeypress (event){
    if (event.key == 'Enter' || event.key == 'Spacebar'){
        handleButtonClick();
    }
  </script>
  </body>
  </html>
```

The 'div' element is given a role of 'button' and an accessible label using the 'aria-label' attribute. The 'tabindex' attribute ensures keyboard focus, and the Javascript functions handle both mouse clicks and keyboard interactions, making the button accessible to a wider range of users.

Offer a webpage with accessibility issues, and ask students to improve its accessibility by adding ARIA roles and attributes.

Date __/__/____ Page No.:_____

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
        initial-scale=1.0">
  <title> Accessibility . Challange </title>
</head>
<body>

  <h1> Welcome to our Website . </h1>

  <div>
  <label for="username">Username : </label>
  <input type="text" id="username" placeholder="Enter your
                                    username">
  <button onclick="login()">login </button>
</div>

<h2> Latest attribute articles </h2>
<ul>
  <li><a href="article1.html"> Article1. </a></li>
  <li><a href="article2.html"> Article2 </a></li>
  <li><a href="article3.html"> Article3 </a></li>
</ul>

<div>
<p> Contact us : email@Example.com </p>
</div>
```

SAMMAN

Date __/__/____ Page No.:_____

```
<script>
    function login(){
        alert('login button clicked!');
    }
</script>
</body>
</html>
```

Ask students to identify and address the accessibility issues in this webpage by adding ARIA roles and attributes. For Example, consider improving the login form, navigation, and contact information to make them more accessible to users with disabilities.

(8) What is the purpose of the 'DOCTYPE' declaration in HTML, and how does it affect the rendering of a webpage in different browsers?

The '<!DOCTYPE>' declaration in HTML specifies the document type and version being used. It helps browsers to interpret and render the HTML document correctly.
The declaration informs the browser about the HTML version and parsing rules to apply. It ensures consistency in rendering across different browsers by providing a standard for interpretation. Modern web pages typically use '<!DOCTYPE html>' for HTML5, which is widely supported and ensures consistent rendering.

Date __/__/____ Page No.:_____

Exercise - Ask students to create a simple HTML document and experiment with different '<!DOCTYPE>' declarations to observe how they affect the rendering in various browsers.

Sure, here's a simple prompt for students to experiment with different '<!DOCTYPE>' declarations in HTML.

1. Create a basic HTML document with a title, header, and some content.
2. Test the document in different browsers (e.g., Chrome, Firefox, and Safari).
3. Experiment with different '<!DOCTYPE>' declarations, such as HTML5 ('<!DOCTYPE html>'), HTML 4.01, or XHTML 1.0.
4. Observe how the rendering may vary in different browsers and note any differences or issues.
5. Reflect on the importance of using the correct '<!DOCTYPE>' declaration for consistent cross-browser rendering.

This hands-on activity will help students understand the role of the '<!DOCTYPE>' declaration in ensuring proper rendering and compatibility across various browsers.